Design of a 4-bit Magnitude Comparator

Lab L05

Introduction:

Figure 5(a) shows the overall structure of the **4-bit adder** that you designed in Lab L03 (without the subtractor circuit). It adds two 4-bit operands **X** and **Y** and computes their 4-bit sum **Z**; it also accepts a carry in signal and produces a carry out signal. The adder's carry in signal is an input bit that is added to the operand bits in bit position 0 (least significant bit). The adder's carry out signal is the carry out bit from the operand bits in bit position 3 (the most significant bit). The 4-bit adder that you previously designed works for both unsigned numbers as well as two's complement numbers; this is because unsigned addition and two's complement addition are done the same way. Another useful arithmetic component is a **magnitude comparator**. Figure 5(b) shows the overall structure of a 4-bit comparator. It compares two 4bit **unsigned** operands **X** and **Y** and the outcome of the comparison is specified by three 1-bit outputs that indicate whether **X < Y, X = Y, or X > Y**.

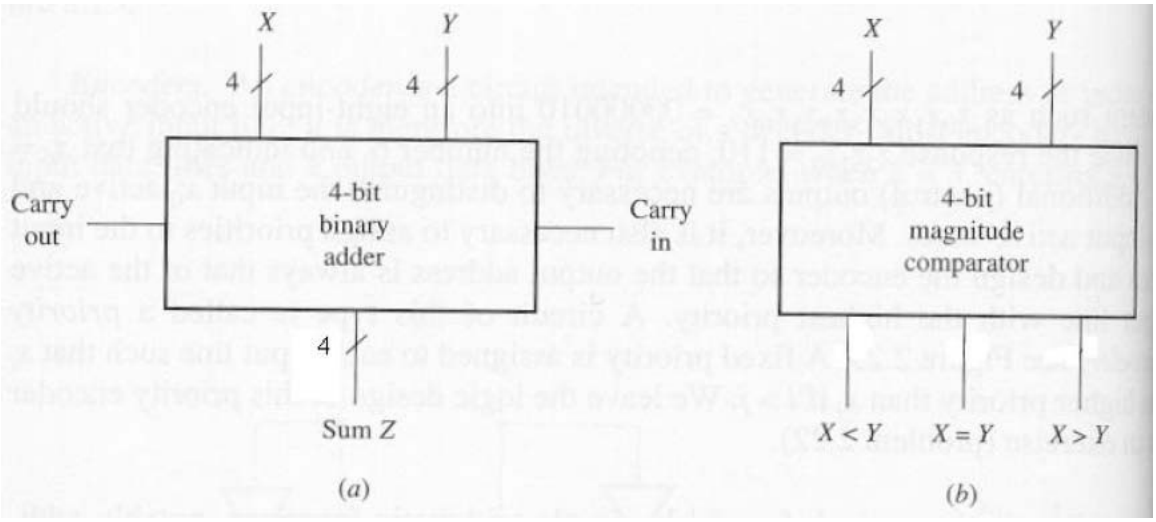

Figure 5: Symbols for (a) a 4-bit adder; (b) a 4-bit magnitude comparator

In Figure 5, each wire that is labeled with a crosshatch represents 4 wires, each carrying one bit of information. The comparator is quite difficult to design at the gate level. In this lab, you will construct a 4-bit magnitude comparator using two 4-bit adder modules and a few two-input logic gates.

Design:

We can design a magnitude comparator for two n-bit unsigned numbers **X** and **Y** efficiently by noting that **X > Y** is equivalent to:

$$\mathbf{X - Y > 0} \quad \text{(i)}$$

Now **Y** can be computed by the subtraction step $\mathbf{(2^n - 1) - NOT(Y)}$, where **NOT(Y)** is the bitwise complement of **Y** and $\mathbf{(2^n - 1)}$ is a sequence of n 1s. For example, if **n = 4** and **Y = 1001** (decimal 9), then **NOT(Y) = 0110** (decimal 6), $\mathbf{2^4 - 1 = 1111}$ (decimal 15), and **Y = 1111 - 0110 = 1001** (decimal 9). Hence inequality (i) can be replaced by $\mathbf{X - (2^n - 1 - NOT(Y)) > 0}$, implying

$$\mathbf{X + NOT(Y) > 2^n - 1 = 11...1} \quad \text{(ii)}$$

Now suppose we add **X** and **NOT(Y)** using an adder such as that of Figure 5(a). If the inequality of (i) is satisfied, then adder's carry-out signal **cout** will be 1, because **X + NOT(Y)** will exceed the largest n-bit number $\mathbf{2^n - 1}$. In the preceding example with **X = 1100** (decimal 12) and **Y = 1001** (decimal 9), we have **X + NOT(Y) = 1100 + 0110 = 10010** (decimal 18), for which the output carry is 1. We can therefore perform the original magnitude test **X > Y** as follows:

1. Compute **NOT(Y)** from **Y.**
2. Add **X** and **NOT(Y)** via an n-bit adder and use the output-carry signal **cout** as the primary output. If **cout = 1**, then **X > Y**; if **cout = 0**, the **X <= Y**.

By switching X and Y, we can generate the "less than" output denoted **X < Y** in exactly the same manner. We do not need the sum outputs of the two adder modules; hence we can discard them, thereby reducing the adders to carry-generation circuits. We have yet to compute the "equal" output denoted **X = Y**. One way to compute it is to compare each bit **Xi** of **X** to the corresponding bit **Yi** of **Y**. But you should come up with an easier way to compute the "equal" output utilizing the **X > Y** and **X < Y** outputs.

Submitting Results

1.All macros are to be submitted with your assignment. Please make sure that each macro includes your name, date and assignment number. Also that they are named correctly ie: LastnameF_L05x (Where Lastname is your last name, F is your first initial and x is a unique letter [a-z] that you used to distinguish each macro.